Research on a Lightweight Multi-Object Detection Algorithm for Robots Used in Tunnel Inspection

Jingze Wu[1,#], Bilie Lou[1,#], Yihang He[2,*]
[1]*Yunnan Yunling Plateau Maintenance Engineering Co., Ltd., Kunming, Yunnan, China*
[2]*Southwest Jiaotong University, Chengdu, Sichuan, China*
[#]*Jingze Wu and Bilie Lou contributed equally to this work*
**Corresponding Author*

Abstract: Tunnel inspection robots face multiple challenges during locomotion, including body vibration, low illumination, and constrained computational resources, which impose dual requirements of lightweight design and high robustness on onboard object detection algorithms. To address these issues, a lightweight multi-object detection algorithm based on improved YOLOv11n is proposed. First, the KT-GLU gated attention enhancement module is introduced, which utilizes gated linear units and depthwise separable convolution to adaptively suppress noise features caused by motion jitter while improving CPU inference efficiency. Second, the KTF-trans attention mechanism is designed, employing asymmetric convolution decomposition and residual connections to enhance directional feature extraction in low-texture tunnel environments. Finally, the SSW-detect lightweight detection head is proposed, which reduces parameter redundancy and improves decision accuracy through cross-scale parameter sharing and group normalization. Experiments are conducted on three public datasets: KITTI, BDD100K-mini, and UA-DETRAC-G2. Results show that the improved model achieves mean average precision (mAP), precision (P), and recall (R) of 63.36%, 66.69%, and 59.31% respectively, representing improvements of 4.6%, 7.25%, and 1.59% over the baseline YOLOv11n, while reducing computational cost to 6.2 GFLOPs. The proposed model effectively balances detection accuracy and computational efficiency, providing a feasible technical solution for real-time object perception on tunnel inspection robots.

Keywords: Tunnel Inspection; Object Detection; YOLOv11; Lightweight Model; Attention Mechanism

1. Introduction

With the deep integration of artificial intelligence and robotics, robots have become a key platform for autonomous inspection missions in complex environments due to their exceptional terrain adaptability and mobility. Compared to wheeled or tracked mobile platforms, robots can traverse unstructured obstacles such as steps, pipes, and loose gravel, demonstrating unique advantages in narrow, confined spaces such as tunnels, utility tunnels, and mine shafts. In recent years, robot products such as Boston Dynamics' Spot and Unitree's Unitree series have become increasingly mature and are gradually being deployed in fields such as industrial inspection, security patrols, and emergency rescue. Integrating computer vision technology with robot platforms to build an integrated inspection system featuring "mobile sensing + intelligent analysis" has become a key development direction for the intelligent operation and maintenance of tunnel infrastructure.

In tunnel inspection scenarios, robots must possess the ability to perceive their surroundings in real time to ensure their own safe passage and to carry out inspection tasks. Specifically, the perception system must simultaneously perform two types of visual tasks: first, the detection and recognition of moving objects within the tunnel—including construction workers, passing vehicles, and motorcycles—to enable autonomous obstacle avoidance and safe navigation, second, the detection of structural defects in the tunnel, such as cracks in the lining, water seepage, and spalling. Among these, the detection of moving objects is a prerequisite for ensuring the robot dog's normal movement and is the primary focus of this study.

Object detection is one of the core tasks in the field of computer vision, aimed at locating and identifying objects of specific categories in images or videos. In recent years, with the rapid advancement of deep learning technologies, object detection methods based on convolutional neural networks (CNNs) have achieved significant success. Girshick et al. proposed the groundbreaking R-CNN [1] model, which successfully applied CNNs to the object detection task for the first time. By generating candidate regions through selective search and then utilizing CNNs for feature extraction, it significantly improved detection accuracy. Subsequently, Girshick et al. introduced Fast R-CNN [2], which achieved feature sharing across the entire image by incorporating a Region of Interest (RoI) Pooling layer, thereby greatly accelerating both training and detection speeds. Ren et al. further proposed Faster R-CNN [3], whose core innovation was the introduction of a Region Proposal Network (RPN). By integrating candidate region generation into end-to-end training, it became the foundational framework for two-stage detectors and has been adopted and refined by numerous subsequent works. Liu et al. proposed the SSD model [4]. SSD performs predictions on feature maps at multiple scales and employs default boxes with varying aspect ratios, effectively improving detection performance for objects of different sizes, particularly excelling in the detection of small and medium-sized objects. Redmon et al. proposed the YOLO series of models. Their core idea is to treat object detection as a regression problem, dividing the image into a grid where each grid cell directly predicts a bounding box and class probability. The YOLO series has gained widespread popularity due to its excellent balance of speed and accuracy, leading to the development of subsequent versions such as YOLOv2, YOLOv3, and YOLOv4 [5].

To address the aforementioned issues, this paper proposes a lightweight improved algorithm based on YOLOv11n [6]. The main contributions are as follows: (1) The introduction of the KT-GLU gated attention enhancement module, which utilizes a gating mechanism and deep separable convolutions to achieve adaptive suppression of noise features, thereby mitigating the interference of motion jitter on detection accuracy,(2) Designing the KTF-trans attention mechanism, which employs asymmetric convolutional decomposition to enhance directional feature extraction capabilities, thereby improving object detection performance in low-texture tunnel environments, (3) Proposing the SSW-detect lightweight detection head, which reduces computational overhead while maintaining detection accuracy through cross-scale parameter sharing and group normalization strategies.

Currently, publicly available visual detection datasets for dynamic targets (such as vehicles and pedestrians) within tunnels remain scarce. Existing tunnel-related datasets are either limited in scale or focused on structural defect detection, making it difficult to support the thorough training and evaluation of deep learning models. In light of this, this paper employs a road scene dataset with a high degree of overlap in target categories to construct an evaluation framework, with a focus on verifying the model's ability to balance accuracy and efficiency under mobile deployment conditions.

2. YOLOv11 Algorithm Improvements

2.1 YOLO-KT Architecture

YOLOv11 is a single-stage object detection algorithm developed by Ultralytics. The YOLO architecture consists of three basic components. First, a convolutional neural network (CNN) processes the raw image data in the backbone module, converting it into multi-scale feature maps, which serve as the primary feature extraction unit. Second, the Neck component acts as an intermediate processing stage, using specialized layers to aggregate and enhance feature representations at different scales. Third, the head component serves as the prediction mechanism, generating the final object localization and classification outputs based on the refined feature maps.

This paper proposes improvements to YOLOv11, whose architecture is shown in Figure 1. By introducing KTNET to reconstruct the backbone network, we enhance CPU inference speed, reduce the number of parameters, and perform information suppression through gated linear units and deep separable convolutions. Furthermore, we design the KTF-trans attention mechanism and implement modifications to the MLP layers and residual connections, resulting in a significant improvement in accuracy. Subsequently, we propose the SSW-detect detection head, which utilizes shared

convolutions and an adaptive scaling mechanism to significantly improve both parameter efficiency and detection accuracy. The following sections provide a detailed introduction to the three components: the KT-GLU gated attention enhancement module, the KTF-trans attention mechanism, and the SSW-detect detection head.

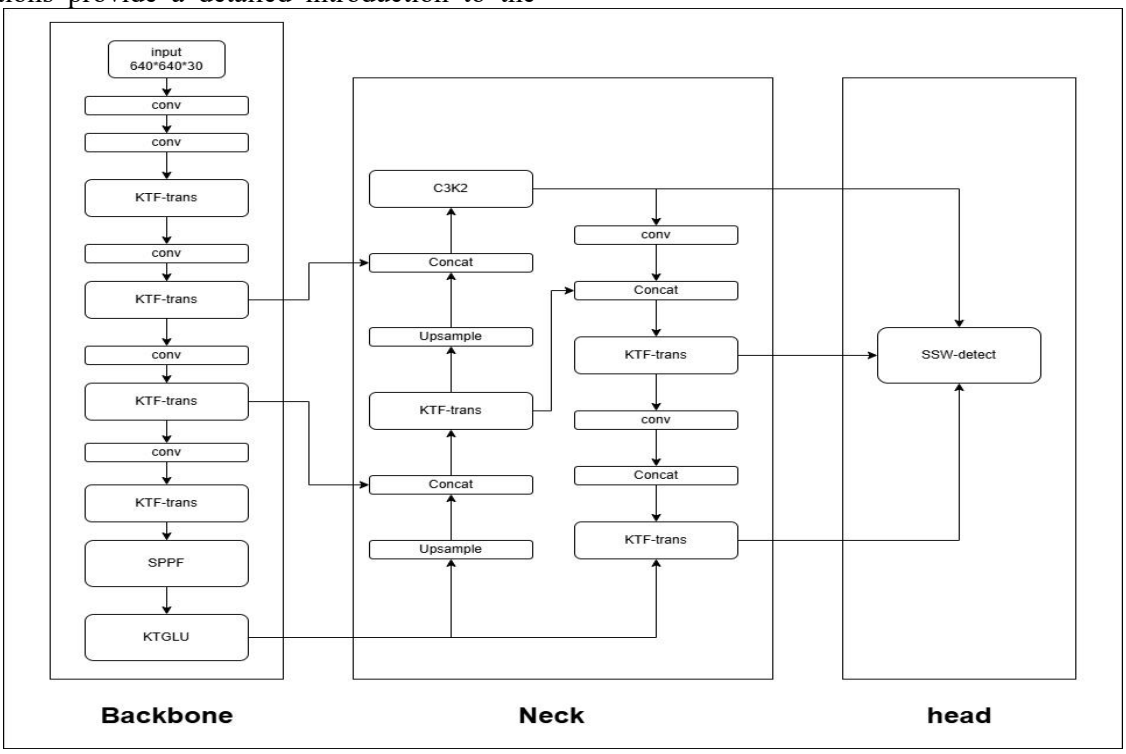


Figure 1. YOLOv11-KT Architecture Diagram

2.2 Algorithm Improvements

2.2.1 KT-GLU gated attention enhancement module

To address the issues of image jitter and motion blur caused by the movement of robots, the KT-GLU module employs a gating mechanism to achieve adaptive suppression of noise features. The module utilizes a branching feature processing strategy: one branch preserves the integrity of the original information, while the other performs selective feature enhancement using position-sensitive attention and convolutional gated linear units, effectively filtering out interference signals introduced by body vibrations.

The KT-GLU module primarily addresses the trade-off between computational efficiency and expressive power in traditional feature extraction modules. The core innovation of this module lies in its adoption of a branched feature processing strategy, which uses 1×1 convolutions to expand the input features and split them into two paths: one preserves the integrity of the original feature information, while the other undergoes a carefully designed enhancement process. This design avoids redundant computations on all features, significantly reducing computational overhead while maintaining performance.

The location-sensitive attention mechanism in the enhanced branch is the first core component of the module. Through a dimension-adaptive query-key-value generation strategy, this mechanism reduces the computational complexity of attention from $O(C^2)$ to $O(C \times C/2)$, where C is the number of channels. In the specific implementation, the dimensions of the queries and keys are set to half that of the head dimension, thereby maintaining sufficient expressive power while effectively controlling the parameter scale. The calculation of attention weights combines spatial position information with inter-channel correlations, and the model's ability to understand spatial structures is enhanced through learnable position encodings.

The second innovation of the module is the introduction of the Convolutional Gated Linear Unit. Unlike traditional multi-layer perceptrons, this unit adopts a processing paradigm of separation, modulation, and fusion. Input features are first expanded and partitioned into two subspaces. One subspace undergoes deep convolution for local feature extraction, with the GELU activation function introducing the

necessary nonlinearity, the other subspace serves as a gating signal, enabling selective feature activation through element-wise multiplication. This gating mechanism allows the network to adaptively adjust the importance of different feature channels, thereby enhancing the model's discriminative power.

The training stability of the entire KT-GLU module is ensured through a multi-level residual design. In addition to standard cross-module residual connections, local residual paths are also integrated within the convolutional gating units, forming a nested gradient propagation structure. This design not only mitigates the vanishing gradient problem in deep networks but also enhances the model's robustness to input perturbations.

From the perspective of computational complexity, the KT-GLU module employs a channel-separated strategy to transform full-channel processing into partial-channel processing, which theoretically reduces the computational load. Additionally, the introduction of a gating mechanism makes feature activations sparser, further improving inference efficiency. While maintaining comparable accuracy, this module demonstrates a better speed-accuracy trade-off compared to traditional self-attention modules, making it particularly suitable for computer vision tasks with high real-time requirements.

The mathematical modeling process for the KT-GLU module is as follows:

First, perform feature transformation on the input:

$$\mathcal{T}_i^{KT-GLU}(X_i) = \varphi_i^{1\times1}(X_i) = [A_i, B_i] \quad (1)$$

$\mathcal{T}_i^{KT-GLU}(X_i)$ denotes the application of the KT-GLU feature transformation to the feature map of layer i, and $\varphi_i^{1\times1}(X_i)$ denotes a 1×1 convolutional expansion of the i-th feature map, which expands the number of input channels from C to 2C, and then splits the map into two branches of equal dimensions along the channel dimension:

A_i Branch: Direct branching, preserving the original feature information.

B_i Branch: Process the branch, enter the attention and gating units.

Then perform the position-sensitive attention calculation:

$$F_i = A_t(B_i) = Softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i + P_e(V_i) \quad (2)$$

A_t denotes the position-sensitive attention function, which enhances attention for branch B_i, $Q_i K_i^T V_i$ represent the query, key, and value matrices, respectively, generated from B_i via 1×1 convolutions, d_k denotes the dimension of the key vector, and $P_e(\cdot)$ denotes the position encoding implemented via 3×3 group convolution.

This is followed by processing using convolutional gated linear units:

$$G_i = C_{GLU}(F_i) = \sigma(\Lambda_i^{3\times3}(U_i)) \odot V_i + F_i \quad (3)$$

Here, C_{GLU} represents a convolutional gated linear unit, F_i is expanded via a 1×1 convolution and split into [U_i , V_i], $\Lambda_i^{3\times3}$ performs a 3×3 deep separable convolution on U_i , $\sigma(\cdot)$ represents the GELU activation function, $\odot$ performs an element-wise multiplication, and finally, a residual connection F_i is added.

Final Output Merging:

$$Y_i = \psi_i^{1\times1}([A_i \oplus G_i]) \quad (4)$$

A_i represents the direct feedforward branch from the input split, G_i represents the enhanced branch following attention and gating, $\psi_i^{1\times1}$ represents the 1×1 convolutional mapping, $\oplus$ represents the channel-dimension concatenation operation.

2.2.2 KTF-trans attention mechanism

In tunnel environments, low-light conditions and areas with weak textures often result in indistinct target features, and the square receptive fields of traditional convolutions struggle to effectively capture directional structural information in narrow, elongated tunnels. The KTF-trans module employs asymmetric convolution decomposition (7×1 and 1×7) to enhance the perception of horizontal and vertical feature patterns while maintaining the receptive field, making it particularly suitable for identifying longitudinally extending structural features and laterally distributed object arrangements within tunnels.

The KTF-trans module is the core feature transformation unit in KTNet. Its design philosophy is to enhance the expressive power of features through convolutional attention mechanisms and multi-layer perceptrons, while further optimizing feature propagation and learning via asymmetric convolutional kernels and residual connections. The KTF-trans module first performs batch normalization on the input features to ensure the stability of the feature distribution, and then processes the features through two parallel convolutional branches: The first branch uses a 7×1 convolutional kernel

(padding=(3,0)) to specifically capture spatial dependencies in the horizontal direction, while the second branch uses a 1×7 convolutional kernel (padding=(0,3)) to specifically capture spatial dependencies in the vertical direction. Compared to traditional 7×7 convolutional kernels, this asymmetric convolutional design maintains the same receptive field while significantly reducing the number of parameters and computational complexity, thereby effectively improving computational efficiency.

The processing flow is the same for each branch: first, feature convolution is performed using F.conv2d, then the resulting feature map is reshaped into a shape of (batch_size, num_heads, channels_per_head, height×width) so that the features can be grouped and processed according to the attention heads. Next, the Softmax activation function is applied in the spatial dimension to generate attention weights, and normalization is performed to ensure the stability of the weight distribution. Finally, the attention weights are reshaped back to the dimensions of the original feature map. The outputs from the two branches are fused through element-wise addition, and then the attention features are mapped back to the original channel space via transposed convolution. This achieves a comprehensive modeling of feature dependencies in both horizontal and vertical directions. This structured directional attention strategy retains the advantages of a global receptive field while breaking away from the uniform processing of traditional convolutions across all directions, enabling the model to capture feature patterns in different directions more effectively.

After convolutional attention processing, the features enter the MLP enhancement module, which consists of a feedforward network formed by two layers of 3×3 convolutions. The first convolutional layer expands the number of input channels to the hidden dimension and introduces a nonlinear transformation using the GELU activation function. The second convolutional layer maps the features back to the output dimension. Dropout is applied between both layers for regularization. Batch normalization is performed before and after the entire MLP processing to ensure the stability of the feature distribution and the smooth flow of gradients. Subsequently, feature fusion is achieved through a dual residual connection strategy: First, a residual connection is added after the convolutional attention module, and then another residual connection is added after the MLP module, with the introduction of a DropPath mechanism for random path inactivation. This design not only alleviates the vanishing gradient problem but also enhances the network's generalization ability.

By combining asymmetric convolutional attention with a multi-layer perceptron, the KTF-trans module innovatively enhances the model's ability to capture local features and long-range dependencies. At the same time, it improves the expressive power of feature maps through directional decomposition, with a particular focus on expanding the receptive field and reducing computational complexity via asymmetric convolutions, thereby demonstrating outstanding performance in feature extraction efficiency.

The specific mathematical modeling process for the KTF-trans module is as follows:

First, perform preprocessing on the input features:

$$H_i = \mathcal{T}_i^{KTF-trans}(X_i) = \varphi_i^{1\times1}(\varphi_i^{norm}(X_i)) \quad (5)$$

$\mathcal{T}_i^{KTF-trans}$ denotes applying a KTF-trans feature transformation to the feature map of layer i, $\varphi_i^{norm}(\cdot)$ denotes applying SyncBatchNorm normalization to the feature map of layer i, $\varphi_i^{1\times1}$ denotes using a 1×1 convolution to align the channel dimensions, ensuring the numerical stability and dimensional consistency of the input features.

Then perform the separated convolutional attention calculation:

$$F_i = A_s(H_i) = A_h(H_i) + A_v(H_i) \quad (6)$$

A_s denotes a split attention function, comprising two branches: A_h attention and A_v attention.

$$A_h(H_i) = Conv_{transpose}^{7\times1}(S_{multi}(Conv^{7\times1}(H_i))) \quad (7)$$

$$A_v(H_i) = Conv_{transpose}^{1\times7}(S_{multi}(Conv^{1\times7}(H_i))) \quad (8)$$

$Conv^{1\times7}$ and $Conv^{7\times1}$ represent 1×7 and 7×1 convolution kernels, respectively, used to capture spatial dependencies in the horizontal and vertical directions, $Conv_{transpose}$ denotes the transposed convolution reconstruction, S_{multi} denotes the multi-head attention activation function:

$$S_{multi}(X) = Reshape(\sigma(Reshape(X, [B, H, C/H, HW]))/(\sum c + \varepsilon), [B, C, H, W]) \quad (9)$$

H represents the number of attention heads, σ represents the softmax function, $\sum c$ represents the sum over channel dimensions, and ε is a fixed numerical constant of 1e-6.

Next is the MLP feature transformation process:

$$M_i = \psi_{MLP}(F_i) = \varphi_2^{3\times3}(D(GELU(\varphi_1^{3\times3}(\varphi^{norm}(F_i))))) \quad (10)$$

ψ_{MLP} represents the MLP transformation module, $\varphi_1^{3\times3}$ and $\varphi_2^{3\times3}$ represent the first and second 3×3 convolutional layers, respectively, $GELU(\cdot)$ represents the Gaussian error linear activation function, and $D(\cdot)$ represents the dropout regularization operation.

Finally, the double residual link output:

$$Y_i = \varphi_i^{1\times1}(\varphi_i^{norm}(X_i)) + DropPath(A_h(H_i) + A_v(H_i)) + DropPath(\psi_{MLP}(A_h(H_i) + A_v(H_i)) \quad (11)$$

$DropPath(\cdot)$ denotes the random path dropout regularization mechanism.

2.2.3 SSW_detect detection head

The computing power of edge computing platforms for robots is limited, and the independent parameter design of traditional detection heads leads to redundant computations. SSW-detect significantly reduces the number of parameters and computational overhead in detection heads by sharing convolutional kernel parameters across scales. Additionally, by replacing batch normalization with group normalization, it reduces dependence on batch size and improves stability during deployment with small batches.

The SSW_Detect (Shared Structure Weighted Detection) detector is a lightweight shared convolutional detection module in KTNet. It is designed to enhance detection accuracy through parameter sharing mechanisms and group normalization strategies, while further optimizing detection performance and computational efficiency via a lightweight convolutional architecture and multi-scale feature fusion. The SSW_Detect detection head first receives feature maps from the backbone at three scales: P3, P4, and P5. These feature maps correspond to downsampling levels of 8×, 16×, and 32×, respectively, and are capable of detecting objects approximately 8×8, 16×16, and 32×32 pixels or larger. To address the detection requirements for objects of different scales, SSW_Detect employs a strategy of sharing convolutional parameters. By reusing the same convolutional kernel weights across multiple detection scales, it significantly reduces the number of model parameters and computational overhead while maintaining the ability to effectively process features at different scales.

The core processing workflow of the detection head employs a lightweight convolutional architecture: first, group normalization replaces traditional batch normalization, dividing the channel dimension into multiple groups for independent normalization. This design not only reduces dependence on batch size but also enhances stability during training with small batches. Next, feature extraction is performed using 3×3 convolutions. These convolutions employ group-wise operations for parameter sharing, allowing convolutional kernels within the same group to be reused across different detection scales. This approach maintains the effectiveness of feature extraction while controlling parameter growth. Subsequently, a 1×1 convolution maps the features to the final detection output dimensions, including classification confidence and bounding box regression parameters. The group normalization and parameter sharing strategies employed throughout the process enable the detection head to significantly reduce computational complexity while maintaining detection accuracy.

The SSW_Detect detection head employs a unified feature channel management strategy for multi-scale feature processing. By uniformly adjusting the number of channels in the P3, P4, P5 feature maps to 256 dimensions, ensuring consistent processing of features at different scales within the detection head. Simultaneously, it employs the DFL (Distribution Focal Loss) mechanism for bounding box regression, transforming traditional direct regression into a distribution prediction problem, thereby improving localization accuracy—particularly for small objects. Finally, three parallel detection branches process object detection tasks at different scales, Each branch employs the same network architecture but processes feature maps at different resolutions. This parallel design allows targets of different scales to be recalled independently, avoiding gradient conflicts between large and small targets. Consequently, it maximizes detection accuracy for multi-scale targets while maintaining manageable computational complexity.

By combining parameter sharing with group normalization, the SSW_Detect detection head innovatively enhances the model's ability to

balance detection efficiency and accuracy. At the same time, its lightweight design improves computational efficiency, with a particular focus on reducing parameter redundancy through shared architecture while maintaining multi-scale detection performance. As a result, it delivers outstanding performance in object detection tasks. Compared to traditional designs with independent detection heads, SSW_Detect maintains detection accuracy while reducing computational overhead through its parameter-sharing strategy, making it suitable for application scenarios with limited computational resources that require high-precision detection.

The mathematical modeling process for the SSW_Detect sensor is as follows:

Let the input multiscale feature map be:

$$X = \{X_{\mathrm{p3}}, X_{\mathrm{p4}}, X_{\mathrm{p5}}\} \quad (12)$$

$X_{Pi} \in \mathbb{R}^{(B\times C_i\times H_i\times W_i)}$ denotes the feature map of the i-th layer, where B is the batch size, C_i is the number of feature channels in the i-th layer, and $H_i \times W_i$ represents the dimensions of the feature map of the i-th layer.

Each input is then subjected to a reparameterized feature transformation:

$$H_i = \Gamma_i^{DBB}(X_i) = \psi_i^{3\times3}(X_i) + \Psi_i^{1\times1}(X_i) + IDE(X_i) + \Psi_i^{1\times1}(\psi_i^{3\times3}(X_i)) \quad (13)$$

$\Gamma_{\mathrm{i}}^{\mathrm{DBB}}$ denotes a re-parameterized feature transformation of the feature map from layer i, $\psi_{\mathrm{i}}^{1\times1}$ denotes a 1×1 convolution of the feature map from layer i, $\psi_{\mathrm{i}}^{3\times3}$ denotes a 3×3 convolution of the feature map from layer i, IDE(·) indicates that only the number of channels of the feature map is processed, with no processing applied to the actual data.

Next, proceed with shared feature extraction:

$$\mathrm{F_i} = \mathrm{M_\tau(H_i)} = \Psi_\rho^{1\times1}(\Lambda_\rho^{3\times3}(H_i)) \quad (14)$$

$\mathrm{M_\tau}$ represents a feature extraction function that shares features, $\Psi_\rho^{1\times1}$ represents a 1×1 convolution followed by GroupNorm normalization, and $\Lambda_\rho^{3\times3}$ represents a 3×3 deep convolution followed by GroupNorm normalization.

Finally, output the results:

$$Y_i = (\eta_s^i \odot \mathcal{M}_{reg}(F_i)) \oplus \mathcal{M}_{cls}(F_i) \quad (15)$$

η_s^i represents the trainable scaling factor for the i-th layer, $\mathcal{M}_{reg}$ represents the prediction from the regression head, $\mathcal{M}_{cls}$ represents the prediction from the classification head, $\odot$ denotes the element-wise multiplication operator, and $\oplus$ denotes the channel-wise concatenation operator.

Upon rearrangement, the entire process can be expressed by the following formula:

$$Y_i = \eta_s^i \odot \mathcal{M}_{reg}(\mathrm{M_\tau}(\Gamma_i^{DBB}(X_i))) \oplus \mathcal{M}_{cls}(\mathrm{M_\tau}(\Gamma_i^{DBB}(X_i))) \quad (16)$$

3. Experimental Results and Analysis

3.1 Dataset Construction

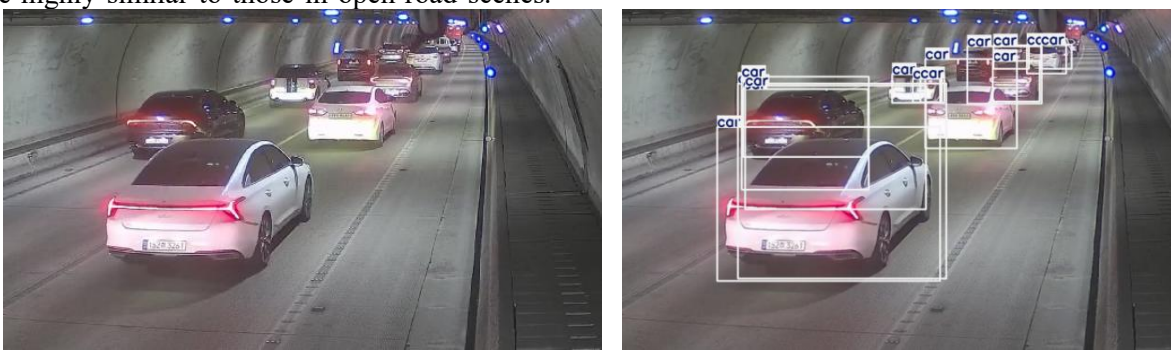
This study selected three public datasets—KITTI [7], BDD100K-mini [8], and UA-DETRAC-G2 [9] — to construct a systematic evaluation framework ranging from standard to extreme scenarios. The KITTI dataset, collected under favorable lighting and weather conditions and featuring high-precision annotations, serves as a performance benchmark for models in ideal environments to assess their fundamental perception capabilities. The BDD100K-mini dataset consists of 10,000 images extracted from the BDD100K dataset, divided into training, test, and validation sets in an 8:1:1 ratio. It covers diverse driving scenarios across different weather conditions, times of day, and geographic regions, with a data distribution that closely mirrors the real world. It is primarily used to evaluate the model's generalization and adaptability in natural, complex environments.

The UA-DETRAC-G2 dataset focuses on simulating rare but critical real-world extreme challenges, including image degradation caused by sensor failures (such as rain streaks and glare) and persistent severe occlusion. These conditions share certain similarities with complex imaging scenarios in tunnels, such as low light and light interference, and can provide a benchmark for evaluating a model's generalization capabilities in tunnel environments.

It should be noted that, due to the scarcity of publicly available datasets for tunnel scenarios, this paper uses the three road scenario datasets mentioned above for model training and testing. These three datasets share a high degree of overlap with tunnel traffic in terms of target classes (vehicles, pedestrians, and motorcycles). This paper focuses on validating the model's ability to balance accuracy and efficiency under mobile deployment conditions, future work will conduct further validation using a custom-built tunnel dataset.

Furthermore, to further validate the model's generalization performance in real-world tunnel

scenarios, this paper uses tunnel scene images from the FastTracker-Benchmark dataset [10] for visual validation. FastTracker-Benchmark is a multi-object tracking benchmark dataset that includes various traffic scenarios such as intersections, tunnels, and highways. As shown in Figure 2, in tunnel sections with good lighting conditions, the visual features of vehicle targets are highly similar to those in open-road scenes. This indirectly validates that models trained on road datasets can generalize to normal tunnel scenarios. However, the real challenges faced by robots for tunnel inspection lie in complex conditions such as low illumination, lighting interference, and motion jitter. Future research should focus on constructing a tunnel dataset that specifically covers these extreme scenarios.



(a)Original Annotated Image (b)Baseline recognition results

Figure 2. Sample Images of the Tunnel Scene from the FastTracker-Benchmark Dataset

3.2 Test Environment and Configuration Settings

The experimental conditions and parameter settings are shown in Table 1 and Table 2, respectively:

Table 1. Experimental Environment Settings

Environment	Specific Value
OP	Ubuntu 20.04
CPU	16 vCPU Intel(R) Xeon(R) Platinum 8481C
GPU	RTX 4090D(24GB) * 1
Deep Learning Framework	Pytorch 2.2.2
Programming Language	Python 3.10
CUDA version	CUDA 12.1

Table 2. Model Training Parameter Settings

Parameter	Value
Optimizer	SGD
Input size	640×640
Lr0	0.01
Epochs	300
mosaic	1.0
weight_decay	0.0005
momentum	0.937
Batch_size	16

3.3 Evaluation Criteria

To comprehensively evaluate the overall performance of the model, this paper uses the following three metrics: MAP [11] (mean average precision) of 50, P [12] (precision), and R [13] (recall), as well as model size and GFLOPS [14] (gigaflops per second). The formulas are shown below:

$$P = \frac{T_p}{T_p+F_p} \quad (17)$$

$$R = \frac{T_p}{T_p+F_n} \quad (18)$$

$$Ap = \int_0^1 P(R)dR \quad (19)$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (20)$$

T_p represents the number of correctly classified target samples, F_p represents the number of misclassified target samples, F_n represents the number of missed target samples, N represents the total number of classes, P(R) denotes the Precision-Recall curve, AP represents the area

under the P(R) curve, mAP represents the mean of the APs across all classes.

3.4 Comparison Results and Analysis

3.4.1 Comparison of the proposed model with baseline models

(a)Original Annotated Image (b)Baseline recognition results (c)Improve model recognition results

Figure 3. Model Recognition Comparison Results

(a)Original Annotated Image (b)Baseline recognition results (c)Improve model recognition results

Figure 4. Model Recognition Comparison Results

(a)Original Annotated Image (b)Baseline recognition results (c)Improve model recognition results

Figure 5. Model Recognition Comparison Results

To provide a more intuitive demonstration of the detection performance of the model described in this paper, Figures 3, 4, and 5 present a comparison of detection results on the test set between YOLOv11 and the improved model.

As shown in Figure 4, the original YOLOv11 model missed several objects, whereas the model proposed in this paper correctly identified these targets. As shown in Figure 3, both the pre-improvement and improved models can correctly detect this object, however, the model proposed in this paper achieves higher detection confidence scores, all of which are higher than those of the pre-improvement model. As shown in Figure 5, the pre-improvement model incorrectly classified the truck object as a car, whereas the improved model correctly identified this object without any false positives, further demonstrating the effectiveness of the improvements made in this paper.

3.4.2 Ablation experiments and analysis

To verify whether the three modules in our model effectively improve upon the original YOLOv11n and to analyze the role of each module, we conducted ablation experiments using the same data and configuration, training for 300 epochs until convergence. The results are shown in Table 3. YOLOv11 denotes the original YOLOv11n model, a represents replacing the C2PSA module in the original YOLOv11 with the KT-GLU module, b represents replacing the detection head in the original YOLOv11 with the SSW_Detect module, and c represents replacing the C3K2 module in the original YOLOv11 with the KT-trans module.

As shown in Table 3, when module a is integrated, the model's precision P, recall R, and mean average precision mAP all improve slightly, while the model size and computational

cost remain virtually unchanged, indicating that KT-GLU enhances the model's discriminative ability, When integrating module B, both the mean average precision (mAP) and recall (R) improved, while the model size and computational requirements (GFLOPS) decreased, further demonstrating the effectiveness of the SSW_Detect detection head, when integrating module C, the model's precision (P) increased by 4.5% and mAP by 2.0%, indicating that Module C plays a crucial role in enhancing the model's accuracy. Combining the three modules significantly improved the model's performance, the mAP of the combined model increased by 4.6% compared to the original model, with precision improving by 7.25% and recall by 1.59%. GFLOPS was reduced by 0.1 GFLOPS. Although the model size increased by 0.8 MB, the improved model significantly boosted mAP while reducing computational requirements, thereby validating the effectiveness of the model improvements described in this paper.

Table 3. Results of the Ablation Experiment

Model	P/%	R/%	mAP/%	Model size/MB	FLOPS/G
YOLOv11	0.5944	0.5772	0.5876	5.2	6.3
v11+a	0.6254	0.6028	0.5997	5.2	6.3
v11+b	0.5452	0.607	0.6027	4.9	5.6
v11+c	0.639	0.571	0.607	6.3	6.9
v11+a+b	0.5909	0.6256	0.603	4.9	5.6
v11+b+c	0.6482	0.5946	0.6247	6.0	6.2
v11+a+b+c	0.6669	0.5931	0.6336	6.0	6.2

3.4.3 Comparative testing and analysis of different models

To demonstrate the effectiveness of the model improvements, we conducted experiments under identical conditions, comparing it with YOLOv9c [15], YOLOv10n, and rtdert-l on three open-source datasets: KITTI, BDD100K_mini, and UA-DETRAC-G2. The experimental results are shown in Table 4 to Table 6.

The experimental results show that, across three open-source datasets, our model demonstrates strong recognition performance compared to models of similar size. Although YOLOv9c outperforms our model on the KITTI and BDD100K_mini datasets, it has 10 times as many parameters and requires nearly 10 times the GFLOPS of our improved model. On the UA-DETRAC-G2 dataset (which includes image degradation scenarios), our model achieved an mAP of 63.36% with only 6.0 MB of parameters, outperforming YOLOv9c (41.3 MB, 63.02%), which has 6.8 times as many parameters. This result demonstrates that our model exhibits superior robustness to low-quality images compared to the baseline methods, which holds significant value for object detection under low-light conditions in tunnels.

The results above demonstrate that the improved model described in this paper maintains strong recognition performance even under constraints on computational power and memory requirements, making it highly suitable for deployment on edge data collection devices used in engineering inspections。

Table 4. Experimental Results for the UA-DETRAC-G2 Dataset

Model	P/%	R/%	mAP/%	Model size/MB	FLOPS/G
YOLOv11	0.5944	0.5772	0.5876	5.2	6.3
YOLOv9c	0.709	0.5697	0.6302	41.3	82.7
YOLOv10n	0.6727	0.5501	0.5856	5.5	6.5
rtdert-l	0.5872	0.5805	0.5554	56.4	100.6
ours	0.6669	0.5931	0.6336	6.0	6.2

Table 5. Experimental Results for the KITTI Dataset

Model	P/%	R/%	mAP/%	Model size/MB	FLOPS/G
YOLOv11	0.8576	0.8	0.8676	5.2	6.3
YOLOv9c	0.9447	0.8861	0.9431	123	82.7
YOLOv10n	0.8374	0.7842	0.8502	5.5	6.5
rtdert-l	0.7251	0.7238	0.7748	56.4	100.6
ours	0.8538	0.8143	0.8785	6.0	6.2

Table 6. BDD100K_mini Dataset Experimental Results

Model	P/%	R/%	mAP/%	Model size/MB	FLOPS/G
YOLOv11	0.5221	0.3886	0.4054	5.2	6.3
YOLOv9c	0.6639	0.47	0.5138	41.3	82.7
YOLOv10n	0.4836	0.3849	0.4069	16.0	6.5
rtdert-l	0.5632	0.337	0.3379	56.4	100.6
ours	0.5322	0.403	0.4228	6.0	6.2

4. Conclusion and Outlook

This paper proposes a lightweight, improved algorithm based on YOLOv11n to address the mobile object detection requirements of robots used for tunnel inspections. To address motion jitter in robotic dogs, a KT-GLU gated attention module is designed to suppress noise, to address the low-texture environment of tunnels, KTF-trans asymmetric convolutional attention is introduced to enhance directional feature extraction, To address computational constraints on edge devices, we propose the SSW-detect shared-parameter detection head to achieve model compression. Experimental results show that the improved model achieves an mAP of 63.36% across three public datasets, representing a 4.6% improvement over the baseline, while reducing computational complexity to 6.2 GFLOPs, demonstrating a good balance between accuracy and efficiency. Future work will focus on the following areas: (1) constructing a real-world tunnel scene dataset containing vehicles, pedestrians, and typical defect samples under various lighting conditions and from multiple viewpoints, (2) deploying the model on a robot platform to conduct on-site tunnel testing and performance evaluation, (3) further exploring model quantization and pruning techniques to reduce the number of parameters and meet the requirements of low-power embedded devices.

References

[1] Girshick R, Donahue J, Darrell T, Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. IEEE Computer Society, 2014:115-126.

[2] Girshick R. Fast r-cnn// Proceedings of the IEEE international conference on computer vision, 2015:1440-1448.

[3] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks//Advances in Neural Information Processing Systems 28 (NIPS 2015). Montreal, Canada: Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2015: 91-99.

[4] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science, 2016:21-37.

[5] Bochkovskiy A, Wang C, Liao H M. YOLOv4: Optimal Speed and Accuracy of Object Detection. Computing Research Repository, 2020, abs /2004.10934.

[6] KHANAM R, HUSSAIN M. YOLOv11: an overview of the key architectural enhancements. KHANAM R, HUSSAIN M. YOLOv11: an overview of the key architectural enhancements. 2410.17725,2024.

[7] Geiger A, Lenz P, Stiller C, et al. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research, 2013, 32(11): 1231-1237.

[8] Yu F, Chen H, Wang X, et al. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 2636-2645.

[9] Wen L, Du D, Cai Z, et al. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding, 2020, 193: 102907.

[10] Hashempoor H, Hwang Y D. FastTracker: Real-Time and Accurate Visual Tracking. KHANAM R, HUSSAIN M. YOLOv11: an overview of the key architectural enhancements.2508.14370, 2025.

[11] Chen J Y, Huang H T, Li Z Y, et al. Feature Enhancement and Metric Optimization for Defect Detection on Steel Surface. Laser & Optoelectronics Progress, 2024, 61(24): 92-101.

[12] Tom Fawcett. An introduction to ROC analysis. Pattern Recognition Letters, 2006, 27(8): 861-874.

[13] Gao Y, Chen P, Liu Z Q, et al. GUAV-YOLO: A Lightweight Detection

Model for Small Target of Unmanned Aerial Vehicles in Grayscale Imag. Acta Optica Sinica, 1-28 [2025-09-25].

[14]Xu J, Chen W C, Jiang M, et al. Lithium Battery X-Ray Defect Detection Based on YOLOv8s. Laser & Optoelectronics Progress, 1-14 [2025-09-25].

[15]WANG C Y, YEH I H, MARK LIAO H Y. YOLOv9: learning what you want to learn using programmable gradient information. European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024:1-21.

